

Software Engineering Fundamentals: MS.NET

C# Types - Arrays

>
accenture

Technology Solutions

Contents

- What is an Array?
- Creating Arrays
 - Declaration
 - Construction
 - Initialization
- Using Arrays
- Stacks
- Queue

Objectives

- At the end of this module you should be able to:
 - Define an array
 - Describe different ways to declare an array
 - Describe different ways to create an array
 - Describe different ways to initialize an array
 - Utilize methods for manipulating an array
 - Demonstrate a single-dimensional array
 - Demonstrate a multi-dimensional array
 - Demonstrate a jagged array (array of arrays)
 - Describe stacks
 - Describe queues

What is an Array?

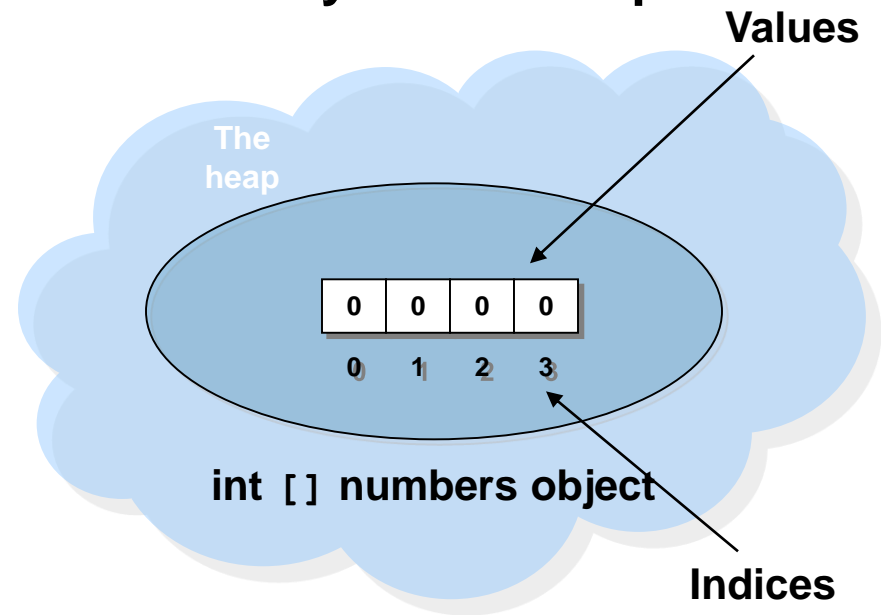
- An array is a data structure that contains a number of variables, which are accessed through computed indices.
- The variables contained in an array, are called the elements of the array. They must all be of the same type, and this type is called the element type of the array.

```
int[] numbers;
```

Diagram illustrating the components of the array declaration:

- `int`: type
- `[]`: Indexing operator
- `numbers`: Identifier name

A one-dimensional array on the heap

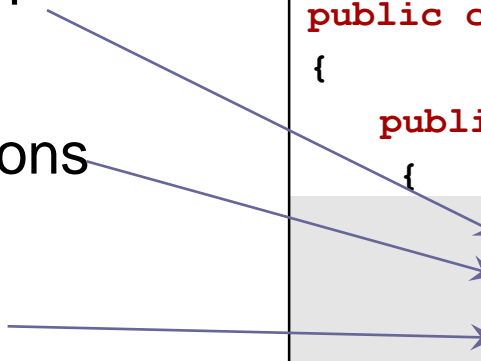


```
int[] numbers;  
numbers = new int[4] ;
```

Creating an Array Declaration

- Declaring an array means providing a name and its data type:
- Single declaration
- Multiple declarations
- Array of objects

```
public class ArrayTest
{
    public static void Main(string[] args)
    {
        int[] numbers;
        char[] letters, symbols;
        string[] countries;
    }
}
```

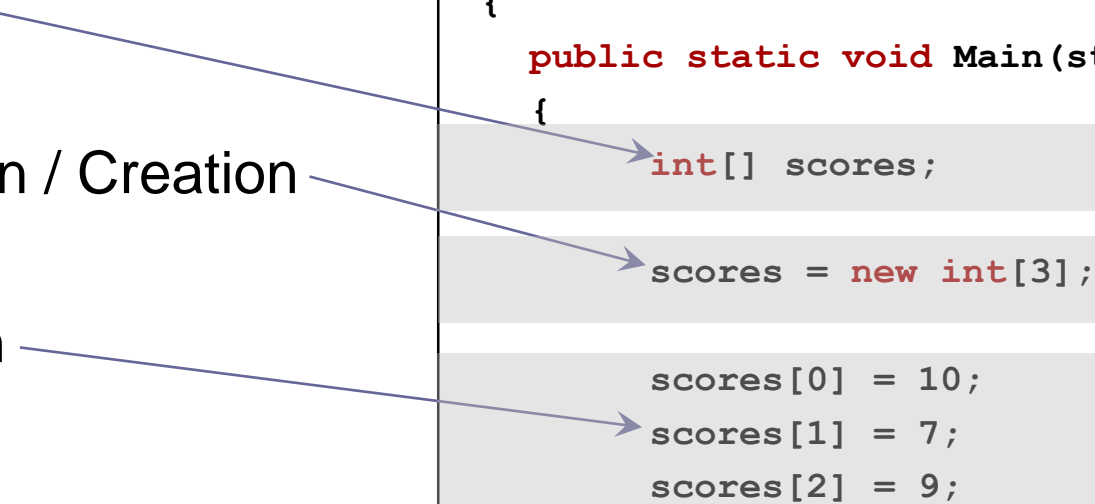


Array Creating Steps

- There are three steps to create an array:

- Declaration
- Construction / Creation
- Initialization

```
public class ArrayTest
{
    public static void Main(string[] args)
    {
        int[] scores;
        scores = new int[3];
        scores[0] = 10;
        scores[1] = 7;
        scores[2] = 9;
    }
}
```



Creating an Array Initialization

- Constructing an array means creating an object of its declared type:
 - Create an array of *int* type consisting of 3 elements
 - Declaration and construction at the same time
- Initializing an array means assigning values to its elements:
 - Array index starts with 0
 - Declaration, construction and initialization at the same time

```
public class ArrayTest
{
    public static void Main(string[] args)
    {
        int[] numbers;
        char[] letters, symbols;
        string[] countries;
        numbers = new int[3];
        string[] currencies = new string[3];

        numbers[0] = 100;
        numbers[1] = 200;
        numbers[2] = 300;
        int[] newNumbers = { 1, 2, 3 };
    }
}
```

Manipulating Arrays

The following are methods for manipulating arrays:

- **Length** gives the size of the array
- Printing each element of an array
- Assigning array (reference) to another array
- Passing array to a method
- Passing anonymous array

Sample Output:

```
100
200
300
6
6
```

```
public class ArrayTest
{
    public static void Main(string[] args)
    {
        int[] numbers = new int[3];
        numbers[0] = 100;
        numbers[1] = 200;
        numbers[2] = 300;
        int[] newNumbers = { 1, 2, 3 };
        for(int i = 0; i < numbers.Length; i++)
        {
            System.Console.WriteLine(numbers[i]);
        }
        numbers = newNumbers;
        System.Console.WriteLine(sumNumbers(numbers));
        System.Console.WriteLine(sumNumbers(
            new int[]{ 3, 2, 1 }));
    }

    static int sumNumbers(int[] n)
    {
        int sum = 0;
        for (int i = 0; i < n.Length; i++)
            sum += n[i];
        return sum;
    }
}
```


Single-Dimensional Arrays

- Single-Dimensional Arrays

- Are declared as

```
int[] numbers;  
numbers = new int[4];
```

- Can also be initialized as

```
string[] myStringArray;  
myStringArray = new string[4];
```

```
int[] numbers = { 1, 3, 5, 7, 9 };
```

```
string[] days = { "Mon", "Tue", "Wed",  
                  "Thu", "Fri", "Sat", "Sun" };
```

- Note: You can omit the **new** operator from the above example. You can assign these values directly without using the new operator.

Multi-Dimensional Arrays

- Multi-Dimensional Arrays

- Are declared as

```
int[,] numbers;  
numbers = new int[4, 2];
```

Two-Dimensional Arrays

- Can also be initialized as

```
int[,,] numbers;  
numbers = new int[4, 2, 3];
```

Three-Dimensional Arrays

```
int[,] numbers =  
{{1,2},{3,4},{5,6},{7,8}};
```

Two-Dimensional Arrays

Jagged Arrays

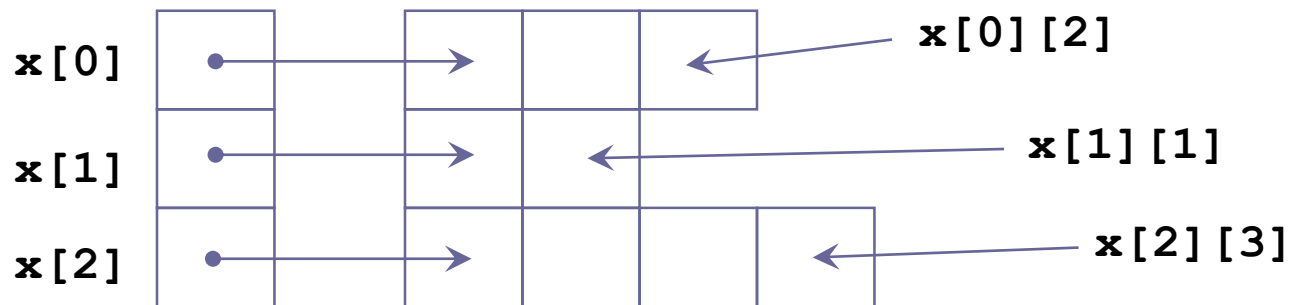
- A jagged array is an array whose elements are arrays. The elements of a jagged array can be of different dimensions and sizes.
- Jagged arrays are also known as an array of arrays

```
int[][] x = new int[3][];
```

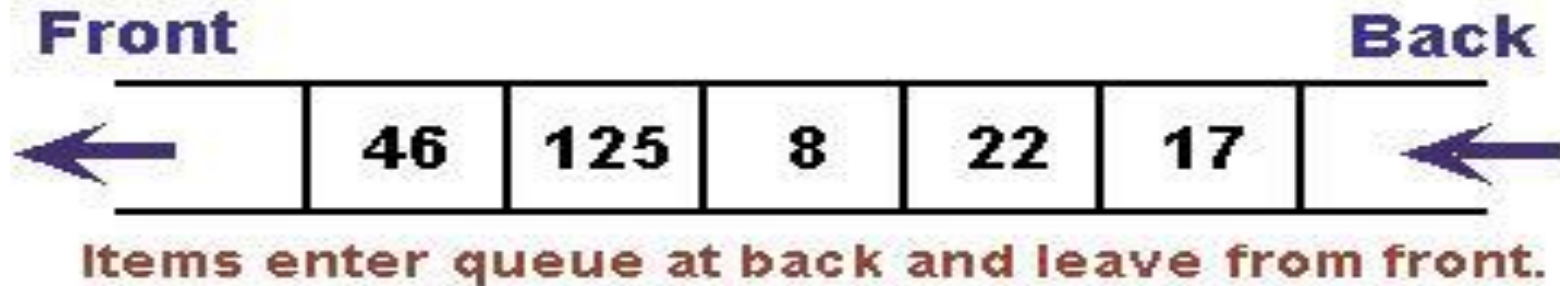
```
x[0] = new int[3]; // first row has three elements
```

```
x[1] = new int[2]; // second row has two elements
```

```
x[2] = new int[4]; // third row has four elements
```



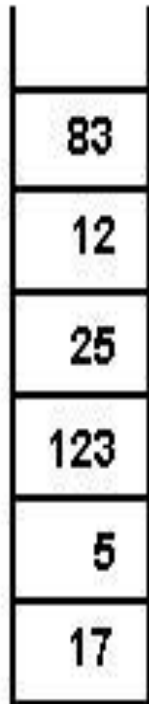
FIFO Queues



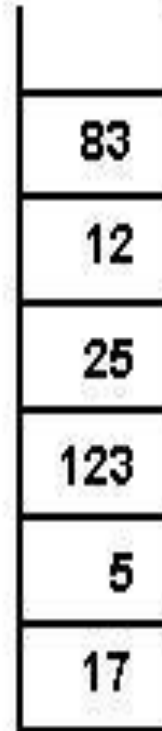
Stacks



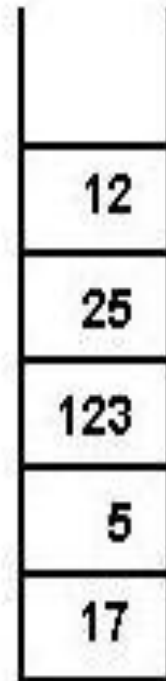
Before push(83)



After push(83)



Before pop(83)



After pop(83)

Key Points

- An array is a collection of values (either primitive or objects) all of which have the same data type
- There are 3 steps in creating an array:
 - Declaration
 - Creation
 - Initialization
- Array elements are accessed using an index that starts at 0
- Length is the property of array that returns its size (i.e., number of elements)
- Arrays can be passed to methods or assigned to variables
- Elements of jagged arrays can be of different dimensions and sizes
- Stacks are ordered sequence or list of items, They use LIFO method to push (add) or pop (remove) an data item.
- Queues are ordered sequence or list of items, They use FIFO method to enqueue (add) or dequeue (remove) an data item.

Questions and Comments

- What questions or comments do you have?

